In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
import mlxtend
import os
```

In [2]:

```
#for file path.
print(os.getcwd())
os.chdir('E:\\01_DataScience\\AI')
```

```
C:\Users\Sonu\Untitled Folder\First Tech Intenship
```

In [3]:

```
data=pd.read_excel("cancer_diagnosis.xlsx")
data.head()
```

Out[3]:

	Serial No	PatientId	Age	Gender	Diagnosis Codes	Chief Complaint	Injury Type	D
0	1	842302	23.0	Female	862.0,807.03,821.00,821.10,823.00,8	MVC-P	Blunt Type of	
1	2	842517	44.0	Female	805.07,813.42,802.4,805.6	Fall<15 Ft.	Blunt Type of	(c Cc un
2	3	843786	34.0	Male	952.03,804.12	Fall<15 Ft.	Blunt Type of	Na f Sul ı
3	4	844359	33.0	Female	805.07,807.03,860.0	Pedestrain	Blunt Type of	I ur c
4	5	844981	54.0	Female	808.42,851.80,805.07,807.01	Fall:Standing	Blunt Type of	F; ja

In [4]:

```
# Using and TransactionEncoder object, we can transform this dataset into an array form
at suitable for typical


# machine learning APIs. Via the fit method, the TransactionEncoder learns the unique l
abels in the dataset, and
# via the transform method, it transforms the input dataset (a Python list of lists) in
to a one-hot encoded NumPy boolean array:
```

In [5]:

```
te = TransactionEncoder()
te_ary = te.fit(data).transform(data)
te_ary
```

Out[5]:

```
array([[ True, False, False, ..., False, False, False],
       [False, False, False, ...,  True, False, False],
       [False,  True, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False]])
```

In [6]:

```
# now i filter the diagnosis _code which are greater than 300.
#data = data.loc[[data["Diagnosis Codes"]>=300])
#data
```

In [7]:

```
data= data.rename(columns={'Diagnosis Codes':'dgcode'})
```

In [8]:

```
data.head()
```

Out[8]:

	Serial No	PatientId	Age	Gender	dgcode	Chief Complaint	Injury Type	D
0	1	842302	23.0	Female	862.0,807.03,821.00,821.10,823.00,8	MVC-P	Blunt Type of	
1	2	842517	44.0	Female	805.07,813.42,802.4,805.6	Fall<15 Ft.	Blunt Type of	(c Cc un
2	3	843786	34.0	Male	952.03,804.12	Fall<15 Ft.	Blunt Type of	Na f Sul
3	4	844359	33.0	Female	805.07,807.03,860.0	Pedestrain	Blunt Type of	l ur c
4	5	844981	54.0	Female	808.42,851.80,805.07,807.01	Fall:Standing	Blunt Type of	F: ja

Now Split a Columns With Multiple Values in Python.

In [9]:

```
#data['PatientId'] = data.index+1
data=data.set_index('PatientId').dgcode.str.split(',', expand = True).stack().reset_ind
ex(1, drop = True).reset_index(name = 'Diagnosis Codes')
```

In [10]:

```
data.head(10)
```

Out[10]:

	PatientId	Diagnosis Codes
0	842302	862.0
1	842302	807.03
2	842302	821.00
3	842302	821.10
4	842302	823.00
5	842302	8
6	842517	805.07
7	842517	813.42
8	842517	802.4
9	842517	805.6

In [11]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113 entries, 0 to 112
Data columns (total 2 columns):
PatientId          113 non-null int64
Diagnosis Codes    113 non-null object
dtypes: int64(1), object(1)
memory usage: 1.8+ KB
```

In [12]:

```
#convert object to float(diagnosis code)
data["Diagnosis Codes"] = data["Diagnosis Codes"].convert_objects(convert_numeric=True)
data.head(5)
```

```
C:\Users\Sonu\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureW
arning: convert_objects is deprecated.  To re-infer data dtypes for object
columns, use Series.infer_objects()
For all other conversions use the data-type specific converters pd.to_date
time, pd.to_timedelta and pd.to_numeric.
```

Out[12]:

	PatientId	Diagnosis Codes
0	842302	862.00
1	842302	807.03
2	842302	821.00
3	842302	821.10
4	842302	823.00

In [13]:

```
data = data.loc[data["Diagnosis Codes"]>=300]
data.head(10)
```

Out[13]:

	PatientId	Diagnosis Codes
0	842302	862.00
1	842302	807.03
2	842302	821.00
3	842302	821.10
4	842302	823.00
6	842517	805.07
7	842517	813.42
8	842517	802.40
9	842517	805.60
10	843786	952.03

Remove dublicate dgcode from raw datasets.

In [14]:

```
data=data.T.drop_duplicates().T
data.head(10)
```

Out[14]:

	PatientId	Diagnosis Codes
0	842302.0	862.00
1	842302.0	807.03
2	842302.0	821.00
3	842302.0	821.10
4	842302.0	823.00
6	842517.0	805.07
7	842517.0	813.42
8	842517.0	802.40
9	842517.0	805.60
10	843786.0	952.03

In [15]:

```
#data = data.groupby(['PatientId'])['dgcode'].reset_index()
#data

#data = data.reset_index().rename_axis(None).rename_axis(None, axis=1)
#data.head(10)
```

In [16]:

```
# Split column into multiple rows.
#new_df = pd.DataFrame(data.dgcode.str.split('|').tolist(), index=data.PatientId).stack
()
#new_df = new_data.reset_index([0, 'PatientId'])
#new_df.columns = ['PatientId', "dgcode"]
```

In [17]:

```
data = data.groupby(['PatientId', 'Diagnosis Codes'])
```

In [18]:

```
#group_name.head(10)
```

In [19]:

```
data.size()
```

Out[19]:

PatientId	Diagnosis Codes	
327599.0	847.670	1
	850.330	1
345678.0	831.050	1
	839.020	1
	951.040	1
842302.0	807.030	1
	821.000	1
	821.100	1
	823.000	1
	862.000	1
842517.0	802.400	1
	805.070	1
	805.600	1
	813.420	1
843786.0	804.120	1
	952.030	1
844359.0	805.070	1
	807.030	1
	860.000	1
844981.0	805.070	1
	807.010	1
	808.420	1
	851.800	1
848406.0	807.010	1
	850.200	1
849014.0	874.800	1
	952.000	1
851509.0	806.090	1
	831.010	1
852552.0	801.250	1
		..
3257405.0	884.100	1
3257692.0	801.350	1
	852.550	1
3258213.0	417.620	1
	433.020	1
	806.050	1
3258415.0	805.060	1
	810.020	1
3263665.0	344.000	1
	345.000	1
	806.230	1
	807.000	1
3263701.0	805.440	1
	806.800	1
	853.220	1
	920.900	1
3264693.0	423.200	1
	905.950	1
3264974.0	336.100	1
	847.100	1
3267651.0	347.060	1
	847.000	1
	850.900	1
3270080.0	351.000	1
	805.440	1
	806.000	1
3272166.0	823.000	1

```
           847.000            1
7571018.0  850.333            1
           861.560            1
Length: 109, dtype: int64
```

In [20]:

```
data=data.size().unstack()
```

In [21]:

```
data.head(10)
```

Out[21]:

Diagnosis Codes	336.1	344.0	344.01	344.03	345.0	347.06	351.0	417.62	423.2	433.02	...	875.
PatientId												
327599.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
345678.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
842302.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
842517.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
843786.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
844359.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
844981.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
848406.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
849014.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
851509.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na

10 rows × 87 columns

In [22]:

```
data = data.fillna(0)
```

In [23]:

```
data.head()
```

Out[23]:

Diagnosis Codes	336.1	344.0	344.01	344.03	345.0	347.06	351.0	417.62	423.2	433.02	...	875.
PatientId												
327599.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
345678.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
842302.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
842517.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
843786.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.

5 rows × 87 columns

Now Applying ARM Algorithm.

In [24]:

```
#data = pd.DataFrame(te_ary, columns=te.columns_)

data=data.astype(bool).astype(int)
#fre_code=apriori(data,min_support=0.01,use_colnames=True)
#fre_code
```

In [25]:

```
fre_code = apriori(data, min_support=0.01, use_colnames=True)
fre_code
```

Out[25]:

	support	itemsets
0	0.025	(336.1)
1	0.025	(344.0)
2	0.050	(344.01)
3	0.050	(344.03)
4	0.025	(345.0)
5	0.025	(347.06)
6	0.025	(351.0)
7	0.025	(417.62)
8	0.025	(423.2)
9	0.025	(433.02)
10	0.025	(745.56)
11	0.025	(801.0)
12	0.025	(801.25)
13	0.025	(801.35)
14	0.025	(802.1)
15	0.025	(802.4)
16	0.025	(804.12)
17	0.025	(805.0)
18	0.075	(805.02)
19	0.025	(805.03)
20	0.025	(805.06)
21	0.075	(805.07)
22	0.025	(805.23)
23	0.050	(805.44)
24	0.025	(805.6)
25	0.025	(806.0)
26	0.025	(806.04)
27	0.075	(806.05)
28	0.050	(806.09)
29	0.025	(806.15)
...	...	...
236	0.025	(821.1, 862.0, 807.03)
237	0.025	(823.0, 862.0, 807.03)
238	0.025	(808.0, 861.21, 839.03)
239	0.025	(808.0, 865.02, 839.03)
240	0.025	(808.0, 865.02, 861.21)
241	0.025	(821.1, 821.0, 823.0)

	support	itemsets
242	0.025	(821.1, 821.0, 862.0)
243	0.025	(821.0, 862.0, 823.0)
244	0.025	(821.1, 862.0, 823.0)
245	0.025	(823.8, 823.0, 823.91)
246	0.025	(831.05, 839.02, 951.04)
247	0.025	(839.35, 839.04, 839.02)
248	0.025	(865.02, 861.21, 839.03)
249	0.025	(865.78, 876.0, 847.1)
250	0.025	(850.1, 875.0, 884.1)
251	0.025	(344.0, 345.0, 806.23, 807.0)
252	0.025	(344.01, 806.09, 344.03, 806.04)
253	0.025	(344.03, 851.34, 852.02, 806.05)
254	0.025	(745.56, 876.0, 865.78, 847.1)
255	0.025	(805.07, 802.4, 813.42, 805.6)
256	0.025	(808.42, 851.8, 805.07, 807.01)
257	0.025	(920.9, 805.44, 853.22, 806.8)
258	0.025	(850.1, 875.0, 884.1, 806.15)
259	0.025	(823.0, 821.1, 821.0, 807.03)
260	0.025	(821.1, 821.0, 862.0, 807.03)
261	0.025	(823.0, 821.0, 862.0, 807.03)
262	0.025	(823.0, 821.1, 862.0, 807.03)
263	0.025	(808.0, 865.02, 861.21, 839.03)
264	0.025	(821.1, 821.0, 862.0, 823.0)
265	0.025	(807.03, 821.1, 821.0, 823.0, 862.0)

266 rows × 2 columns

In [26]:

```
type(fre_code)
```

Out[26]:

```
pandas.core.frame.DataFrame
```

In [27]:

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

In [28]:

```
rules = association_rules(fre_code,metric="lift", min_threshold=1)
rules.head()
```

Out[28]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(336.1)	(847.1)	0.025	0.050	0.025	1.0	20.0	0.023750
1	(847.1)	(336.1)	0.050	0.025	0.025	0.5	20.0	0.023750
2	(344.0)	(345.0)	0.025	0.025	0.025	1.0	40.0	0.024375
3	(345.0)	(344.0)	0.025	0.025	0.025	1.0	40.0	0.024375
4	(344.0)	(806.23)	0.025	0.025	0.025	1.0	40.0	0.024375

In [29]:

```
rules['lift'].max()
```

Out[29]:

```
40.0
```

In [30]:

```
rules = association_rules(fre_code,metric="confidence", min_threshold=1)
rules.head()
```

Out[30]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(336.1)	(847.1)	0.025	0.050	0.025	1.0	20.0	0.023750
1	(344.0)	(345.0)	0.025	0.025	0.025	1.0	40.0	0.024375
2	(345.0)	(344.0)	0.025	0.025	0.025	1.0	40.0	0.024375
3	(344.0)	(806.23)	0.025	0.025	0.025	1.0	40.0	0.024375
4	(806.23)	(344.0)	0.025	0.025	0.025	1.0	40.0	0.024375

In [31]:

```
rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
rules.head()
```

Out[31]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(336.1)	(847.1)	0.025	0.050	0.025	1.0	20.0	0.023750
1	(344.0)	(345.0)	0.025	0.025	0.025	1.0	40.0	0.024375
2	(345.0)	(344.0)	0.025	0.025	0.025	1.0	40.0	0.024375
3	(344.0)	(806.23)	0.025	0.025	0.025	1.0	40.0	0.024375
4	(806.23)	(344.0)	0.025	0.025	0.025	1.0	40.0	0.024375

In [32]:

```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

Out[32]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
150	(344.0, 345.0)	(806.23)	0.025	0.025	0.025	1.0	40.000000	0.0
151	(344.0, 806.23)	(345.0)	0.025	0.025	0.025	1.0	40.000000	0.0
152	(345.0, 806.23)	(344.0)	0.025	0.025	0.025	1.0	40.000000	0.0
156	(344.0, 345.0)	(807.0)	0.025	0.025	0.025	1.0	40.000000	0.0
157	(344.0, 807.0)	(345.0)	0.025	0.025	0.025	1.0	40.000000	0.0
158	(345.0, 807.0)	(344.0)	0.025	0.025	0.025	1.0	40.000000	0.0
162	(344.0, 806.23)	(807.0)	0.025	0.025	0.025	1.0	40.000000	0.0
163	(344.0, 807.0)	(806.23)	0.025	0.025	0.025	1.0	40.000000	0.0
164	(806.23, 807.0)	(344.0)	0.025	0.025	0.025	1.0	40.000000	0.0
168	(344.01, 344.03)	(806.04)	0.025	0.025	0.025	1.0	40.000000	0.0
169	(344.01, 806.04)	(344.03)	0.025	0.050	0.025	1.0	20.000000	0.0
170	(344.03, 806.04)	(344.01)	0.025	0.050	0.025	1.0	20.000000	0.0
172	(344.01, 344.03)	(806.09)	0.025	0.050	0.025	1.0	20.000000	0.0
173	(344.01, 806.09)	(344.03)	0.025	0.050	0.025	1.0	20.000000	0.0
174	(344.03, 806.09)	(344.01)	0.025	0.050	0.025	1.0	20.000000	0.0
175	(344.01, 900.03)	(805.02)	0.025	0.075	0.025	1.0	13.333333	0.0
176	(344.01, 805.02)	(900.03)	0.025	0.025	0.025	1.0	40.000000	0.0
177	(900.03, 805.02)	(344.01)	0.025	0.050	0.025	1.0	20.000000	0.0
179	(344.01, 806.09)	(806.04)	0.025	0.025	0.025	1.0	40.000000	0.0
180	(344.01, 806.04)	(806.09)	0.025	0.050	0.025	1.0	20.000000	0.0
181	(806.04, 806.09)	(344.01)	0.025	0.050	0.025	1.0	20.000000	0.0
183	(344.03, 806.09)	(806.04)	0.025	0.025	0.025	1.0	40.000000	0.0
184	(344.03, 806.04)	(806.09)	0.025	0.050	0.025	1.0	20.000000	0.0

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
185	(806.04, 806.09)	(344.03)	0.025	0.050	0.025	1.0	20.000000	0.0
187	(344.03, 851.34)	(806.05)	0.025	0.075	0.025	1.0	13.333333	0.0
188	(344.03, 806.05)	(851.34)	0.025	0.025	0.025	1.0	40.000000	0.0
189	(851.34, 806.05)	(344.03)	0.025	0.050	0.025	1.0	20.000000	0.0
191	(344.03, 852.02)	(806.05)	0.025	0.075	0.025	1.0	13.333333	0.0
192	(344.03, 806.05)	(852.02)	0.025	0.025	0.025	1.0	40.000000	0.0
193	(852.02, 806.05)	(344.03)	0.025	0.050	0.025	1.0	20.000000	0.0
...	...	...	...	...	...	...	...	
596	(821.1, 862.0)	(821.0, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
597	(821.1, 823.0)	(821.0, 862.0)	0.025	0.025	0.025	1.0	40.000000	0.0
598	(821.0, 862.0)	(821.1, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
599	(821.0, 823.0)	(821.1, 862.0)	0.025	0.025	0.025	1.0	40.000000	0.0
600	(862.0, 823.0)	(821.0, 821.1)	0.025	0.025	0.025	1.0	40.000000	0.0
604	(823.0, 821.0, 821.1, 807.03)	(862.0)	0.025	0.025	0.025	1.0	40.000000	0.0
605	(821.0, 821.1, 862.0, 807.03)	(823.0)	0.025	0.075	0.025	1.0	13.333333	0.0
606	(823.0, 821.1, 862.0, 807.03)	(821.0)	0.025	0.025	0.025	1.0	40.000000	0.0
607	(823.0, 821.0, 862.0, 807.03)	(821.1)	0.025	0.025	0.025	1.0	40.000000	0.0
608	(821.0, 821.1, 862.0, 823.0)	(807.03)	0.025	0.050	0.025	1.0	20.000000	0.0
609	(821.0, 821.1, 807.03)	(862.0, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
610	(823.0, 821.1, 807.03)	(821.0, 862.0)	0.025	0.025	0.025	1.0	40.000000	0.0
611	(821.1, 862.0, 807.03)	(821.0, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
612	(823.0, 821.0, 807.03)	(821.1, 862.0)	0.025	0.025	0.025	1.0	40.000000	0.0

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
613	(821.0, 862.0, 807.03)	(821.1, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
614	(823.0, 862.0, 807.03)	(821.0, 821.1)	0.025	0.025	0.025	1.0	40.000000	0.0
615	(821.0, 821.1, 823.0)	(862.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
616	(821.0, 821.1, 862.0)	(823.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
617	(821.1, 862.0, 823.0)	(821.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
618	(821.0, 862.0, 823.0)	(821.1, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
619	(821.1, 807.03)	(821.0, 862.0, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
620	(821.0, 807.03)	(821.1, 862.0, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
621	(823.0, 807.03)	(821.0, 821.1, 862.0)	0.025	0.025	0.025	1.0	40.000000	0.0
622	(862.0, 807.03)	(821.0, 821.1, 823.0)	0.025	0.025	0.025	1.0	40.000000	0.0
623	(821.0, 821.1)	(823.0, 862.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
624	(821.1, 823.0)	(821.0, 862.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
625	(821.1, 862.0)	(823.0, 821.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
626	(821.0, 823.0)	(821.1, 862.0, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
627	(821.0, 862.0)	(823.0, 821.1, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0
628	(862.0, 823.0)	(821.0, 821.1, 807.03)	0.025	0.025	0.025	1.0	40.000000	0.0

329 rows × 10 columns

In [ ]: